PRV
PATENT- OCH REGISTRERINGSVERKET
Patentavdelningen

Intyg
Certificate

Härmed intygas att bifogade kopior överensstämmer med de handlingar som ursprungligen ingivits till Patent- och registreringsverket i nedannämnda ansökan.

This is to certify that the annexed is a true copy of the documents as originally filed with the Patent- and Registration Office in connection with the following patent application.

(71)	*Sökande Applicant (s)*	*Ludesi AB, Lund SE*
(21)	*Patentansökningsnummer Patent application number*	*0201894-3*
(86)	*Ingivningsdatum Date of filing*	*2002-06-19*

Stockholm, 2003-06-25

För Patent- och registreringsverket
For the Patent- and Registration Office

Sonia André

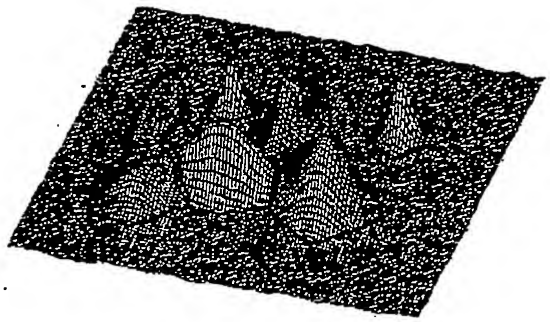
Avgift
Fee

Method for Digital Image Processing

Gustav Wallmark

June 19, 2002

Center of Mathematical Sciences
Mathematics
Lund Institute of Technology



1 Introduction

1.1 Proteomics

The field of proteomics has gained importance over the last ten years. After the human genome was sequenced, in year 2000, the challenge has been to interpret this large amount of information for improving health care and discovering new drugs. This is where proteomics enter the arena.

The term proteome refers to all proteins produced by a living organism, much as the genome is the entire set of genes in an organism. Proteomics indicates proteins expressed by a genome and is the systematic analysis of protein profiles. It was not until 1995 that the actual term proteomics was introduced.

When examining the genome one looks at stationary information that does not change over time. This means that it is difficult to understand which role a certain gene plays in the dynamic process, as living is. On the other hand, the proteome varies over time and is defined as the proteins present in a sample at a certain point in time. Therefore, the proteome will make it possible to indicate which part of the genome that is active in a process. This is why proteomics parallels the related field of genomics.

Proteome research is far more complex than genome research, because the proteins in a cell can not be amplified, its presence is dynamic and its solubility is variable. To overcome this complexity and still get useful information from the analysis, three major steps in proteome research exist. They are the following

1. Separation of individual proteins by two-dimensional polyacrylamide gel electrophoresis.
2. Identification by mass spectrometry or N-terminal sequencing of individual proteins recovered from the gel.
3. Storage, manipulation, and comparison of the data using bioinfomatics.

Naturally, every step mentioned above is equally important, but for further understanding of this master's thesis only the first step is of interest. This part will be briefly introduced below. For more details about proteomics please refer to Jain [1].

1.2 Two-Dimensional Gel Electrophoresis

About 25 years ago the two-dimensional gel electrophoresis (2-DE) was introduced and described by O'Farrell [2] and Klose [3]. Since then, 2-DE has been used in a diverse range of applications, where separation of proteins is essential. In the beginning the technique was rough and it had quite a few pitfalls and difficulties. Over the past years these problems have been partly eliminated by new improved 2-DE techniques and software, but they still exists. Despite this, the 2-DE importance in proteomics has grown and still today it is unparalleled in its ability to separate and array complex proteins.

Before the 2-DE separation technique can be applied, a protein sample has to be extracted from the examined organism. The sample has to be pure and free of other contaminating substances, otherwise the separation will be disturbed or even fail. There exists numerous techniques to purify the proteins and today it is not problematic, see Berkelman [4].

After receiving the protein sample, it is placed on a so called strip. The strip is made of polyacrylamide gel, contains a pH gradient and is about ten centimeters long and one centimeter wide. Because of the pH gradient, the proteins will separate according to their isoelectric points over a period of about ten hours. When this is done the strip and the one-dimensional separated proteins are transferred to a second dimension, according to their molecular weights. The transformation is done by placing the strip on the side of a plate consisting of sodium dodecyl sulfate-polyacrylamide gel. An electric field is applied over the plate and the strip, forcing the proteins to merge into the plate at different speeds depending on their molecular weight. In this way the proteins have been separated both in a pH and a molecular dimension. For a schematic view of the separation process, see Figure 1, below.

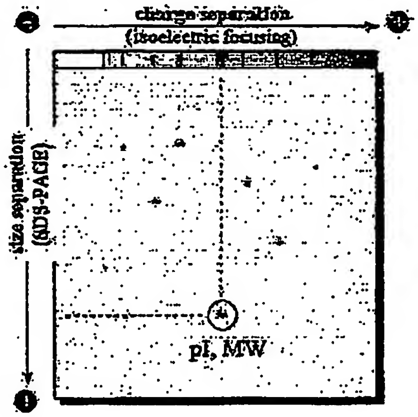


Figure 1: Schematic view of the 2-DE separation

When the separation of the proteins has been completed, the gel has to be stained. This will make the non-colored separated proteins visible to the naked eye. There exist several different staining techniques and substances, which all have their advantages and disadvantages. The common way to stain gels is with a silver solution that colors proteins black, see Berkelman [4].

Finally, the stained gel is inserted into a gel scanner and typically transferred to a gray scale digitized TIF image. This image is fed to a gel software in a computer and further analyzed. For more details about 2-DE techniques and its application please refer to Ong [5].

1.3 Master's Thesis Problem Definition

Digitized images received from 2-DE protein separations are very complex. Often many different types of protein exists in a protein sample, which in turn yields many different protein spots in the image. Typically a number of between 1000 and 3000 protein spots are present and visible in an image.

If a scientist wants to know which type and the amount of proteins a sample contains, the 2-DE image has to be evaluated. To do this manually is very time consuming and practically non-performable. It is estimated that one image requires approximately five hours or more of manual work. In a normal scientific study, with several protein samples, as many as 50 images have to be evaluated. Thus, manual work is impossible if research should be fast and cost effective.

In recent years computer software have been developed to minimize the evaluation time. Even though there exist many software today, quite a few of them are not very reliable and still require many hours of manual work.

1.4 The Goal of This Master's Thesis

The goal of this master's thesis is to investigate the possibility to segmentate protein spots from 2-DE images, with the help of evolving interfaces. The goal is also to automate the segmentation process as much as possible, while containing reliability, accuracy and speed.

1.5 Organization of This Report

The material in this report is divided into ten chapters. In chapter two a closer look at the images to be segmented will be made. Problems, certain types and examples of images will be shown and clarified. Moving on to chapter three, previous work done in 2-DE segmentation is discussed and two examples of well established approaches are given. The theory of image processing is introduced in chapter four. Chapter five explains the methodology of evolving interfaces with the numerical approximation scheme, the Fast Marching Method. In chapter six the implementation of the segmentation system is described. A comparison and evaluation of the proposed segmentation system is conducted in chapter seven. The implementation proposal in this master's thesis is far from complete and suggestions on further improvements are given in chapter eight. Chapter nine concludes this master's thesis and finally chapter ten acknowledges involved persons.

2 A Closer Look at 2-DE Images

In the introduction to this master's thesis the 2-DE technique was briefly described. The technique produces a result consisting of two parts. The first is the gel plate with the physically separated proteins. It can be further investigated, for example, when it is interesting to do an identification by mass spectrometry or N-terminal sequencing of individual proteins. Recall, this is the next step after 2-DE analysis, introduced in Section 1.1. The other part is the digitized image produced with a gel scanner. This image is further analyzed in a software and could be denoted as a bridge between the first step, in Section 1.1, and the two following. Both the gel plate and the 2-DE image are closely connected during a full investigation of the proteome.

2.1 Good Quality 2-DE Images

Only the digitized images are of interest, because the segmentation process is based on these. Therefore, it is of outmost importance that they are of good quality. Good quality means

1. Clear and focused images.
2. Smooth images without noise.
3. Images with as little background variations as possible.
4. Non-saturated images.
5. Evenly spread and well separated proteins throughout the images.
6. Images free from artifacts, such as non-protein patterns.

The quality of the images depends on the protein separation steps, the staining technique and the scanning. In each of these steps there exist several difficulties, which decrease possibilities to end up with a good image. In some cases only a very skilled and experienced 2-DE analyzer will deliver quality images.

2.2 Examples of 2-DE Images

Below, in Figure 2, a typical gel image is shown. It contains approximately 1200 different proteins. In the image some basic features are marked. These are

- *Protein Spot:* Black spots which have a high concentration of certain proteins. The protein spots can vary in size and shape, but they are commonly elliptic or close to circular.
- *Overlapping Spots:* When two protein spots are to close to each other in the image they overlap. This means that their area of distribution overlaps.
- *Background:* Area in the image, which does not contain any protein spots.
- *Varying Background:* Intensity variations of the background.
- *Artifact:* Object which appear in the image and is not a protein spot.

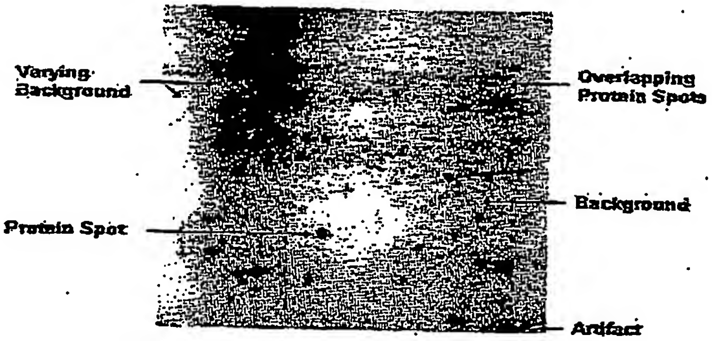


Figure 2: A typical 2-DE image.

Two low quality images are illustrated in Figure 3 and Figure 4. The first image is saturated. A saturated image is often the result when too much staining substance has been used in proportion to the amount of proteins contained in the sample. Protein spots will be cut off and their shape in three dimensions will not be true. This may create difficulties when quantifying proteins in such an image.

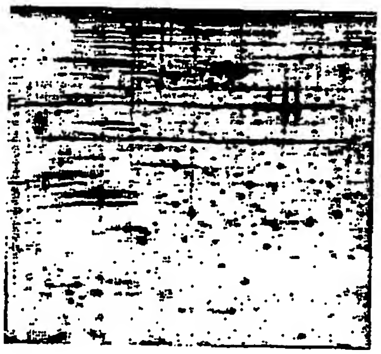


Figure 3: A saturated 2-DE image.

The next image, Figure 4, is of poor quality because it contains too much proteins in the introduced sample. Long stripes exist in the image and cover other interesting proteins. A correct evaluation is difficult to perform. The separation failed when the first dimension was transferred to the second dimension.

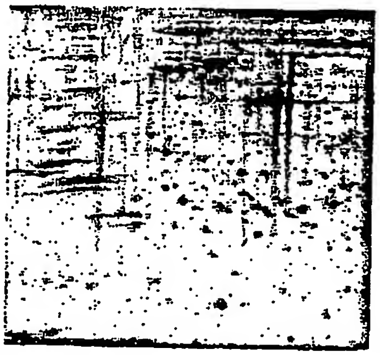


Figure 4: An overloaded 2-DE image.

Over the last couple of years, since the proteomics area began, new improved 2-DE techniques have been developed. Therefore, today it is possible to require good quality images to guarantee correct evaluation results from a segmentation.

3 Previous Work

3.1 Introduction

The research in computer-assisted analysis of two-dimensional gel electrophoresis began about twenty years ago. Mathematical methods for the image processing were based on results obtained in fields such as image processing, computer vision and artificial intelligence. Today, there is a big race between several big companies in 2-DE software business. Yet, none of these companies has won the market and it does not exist any standards in 2-DE image analysis.

In parallel with all these companies, there are several different approaches to perform 2-DE image segmentation. When the development of these approaches began, computer performance was a main issue. Since that time, computers have become tens and hundreds of times more powerful and today the main focus lies on correctness, automation and reliability.

In this section two of the well known approaches to the segmentation will be briefly introduced. For a more complete presentation of these and other used methods, see Pedersen [6]. It is often a combination of different approaches that will lead to good and well performing 2-DE image segmentation.

3.2 Watershed Approach

In the Watershed approach the 2-DE image is regarded as a landscape with hills and valleys. Gray level value determines the height of the landscape. The goal for the Watershed is to divide each valley into a separate region, so that the whole image is divided into a mosaic. In Figure 5, the concept of the Watershed in one dimension is shown.

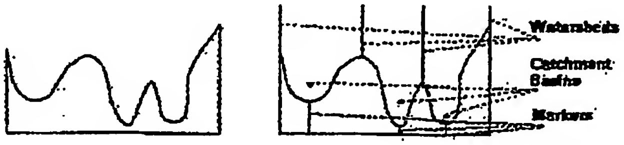


Figure 5: Two figures explaining the watershed concept in one dimension. In each local minimum the water enters and fills each catchment basin. When water from two different basins meet, a dam wall is built.

To create this subdivision the Watershed uses a technique, which could be compared to water flooding. In each valley there exists a local minimum. In the local minima holes are drilled, so that water can flow into the valleys. Continuing, the whole image with the holes is lower into a lake of water. As it is lowered, water will start to flow through the holes and fill the landscape. When water from two different regions meet, a dam wall is built to prevent the water from mixing. As the water level in the landscape rises, more and more walls are raised. Finally, when the whole landscape has been immersed into the lake, a complete subdivision of the image has been formed.

The segmentation problem is now transferred into finding local minima in the image and to decide which regions in the mosaic image that are connected to protein spots.

The main disadvantage with the Watershed approach is the tendency of over segmentation, due to noise in images. This can be avoided by Scale Space Watershed and Marker Controlled watershed. For more details about the Watershed approach see Vincent [7], Pedersen [6] and Wallmark [8].

3.3 Spot Modelling Approach

The assumption in this approach is that the protein spots in an image have some common characteristics that can be captured by a model. The idea is to find parameters that can change the model so it fits with different protein spots. A model $C(x, y, \theta)$ is defined, where x and y denote the position of the model and θ its parameters. The goal is to optimize the model to the image $I(x, y)$, so that the error is minimized. This can be expressed as

$$\tilde{\theta} = arg \min_{\theta} \sum_{x,y \epsilon w} (I(x,y) - C(x,y,\theta))^2 \quad (1)$$

where $x, y \epsilon w$ and w is a region in the image $I(x, y)$. The region w has to be big enough to contain a spot, but small enough to avoid containing multiple spots.

A model commonly used is the Gaussian model given by

$$C_G(x, y, \theta) = B + \alpha e^{-\frac{(x-x_0)^2}{2\sigma_x^2}} e^{-\frac{(y-y_0)^2}{2\sigma_y^2}} \quad (2)$$

This model can be modified with a diffusion equation, but this will not be further investigated here.

Advantages with this approach is that it does not depend on predefined markers to find protein spots in the image. It can also module spots that are saturated and spots that are isolated, in a good way. Difficulties arise when spots overlap, which they do very often in 2-DE images. Instead of creating a module for two small spots one big spot is selected as the best description, because the model is not enough complicated to model two small spots.

For further reading please consult Pedersen [6] and Bettens [9].

4 Theory of Image Processing

4.1 Introduction

The mathematical branch that deals with digital images is called *image processing*. The interest in digital images began in the 1920s, when an image transmitting system was built between London and New York. Ever since, this scientific field has grown tremendously and today it occupies researchers all over the world.

There exist some fundamental steps in image processing that are used by a common image processing system. The first step is the *image acquisition*. Digital monochrome images are represented by a light intensity function $f(x,y)$, where x and y denote spatial coordinates. The value of f at any point (x,y) is proportional to the brightness of the image at that point. A digital image can also represent a color image. There are several different ways of representing color images digitally and one of them is the *RGB-system*. Instead of having a single valued function $f(x,y)$ for a monochrome image, as above, the function becomes three-valued. In each point (x,y) a value is given for the redness, greenness and blueness in the image.

Next, after the digitalized image has been obtained, the *image preprocessing* stage is conducted. In this stage the image is improved to increase the chances for success of other following stages. After the preprocessing stage follows the *image analysis*. Here, the image could be *segmented*, *represented*, *described*, *recognized* and *interpreted*.

Below, some techniques for image preprocessing and image segmentation will be further discussed. A part with image modelling will also be introduced. The interested reader is referred to Gonzalez [10] for further reading about the other parts in image processing.

4.2 Image Preprocessing

To improve the chances of success in a following image processing stage, some image preprocessing is done. Techniques to remove noise in images will be introduced below.

To remove noise in images a filter of some kind can be used. Filtering the image is the same as running a mask through the image. A mask could be treated as a function given by

$$g(x,y) = T[f(x,y)]$$

where $f(x,y)$ is the input image, $g(x,y)$ is the processed image, and T is an operator of f, defined over some neighborhood of (x,y). Normally, the neighborhood about (x,y) is defined by a square or rectangular subimage area centered at (x,y), as Figure 6 shows. The mask is run through an image when the operator T has visited all points (x,y) in the image and thus generated a new modified image $g(x,y)$.

Noise in images is often present as small discontinuities. This means that only one of several pixels in a neighborhood has been disturbed by the noise. If the noise has this property, it can be removed by taking the mean of a small neighborhood around each pixel. The most commonly used mean filter is by far

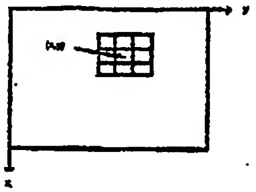


Figure 6: A three by three neighborhood about a point (x, y) in an image.

the Gaussian filter of the form

$$G_\sigma = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3}$$

In Figure 7 two different Gaussian filters with different sigma are shown.

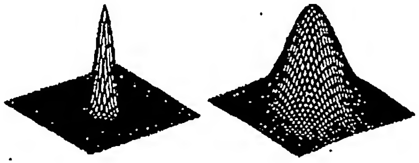


Figure 7: Two Gaussian filters. Left: $\sigma = 10$. Right: $\sigma = 100$.

It is also possible to utilize a simple mask, with ones in each position, as a mean filter. The size of the filter determines the size of the neighborhood to create the mean value from. An example of such a mean filter is given in Figure 8.

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Figure 8: Mean filter with a size of three by three pixels.

4.3 Image Segmentation

An image is said to be segmented when it is subdivided into its constituent parts or objects. The level to which this subdivision is carried out depends on the problem in question. Thus, the segmentation should stop when the object of interest in an image has been isolated from the other non interesting parts or objects. This can be written as

$$\begin{aligned} A, B &\in C \\ A \cap B &= 0 \\ A \cup B &= C \end{aligned}$$

where A symbolizes the objects of interest, B symbolizes the non-interesting objects or parts of the image and C denotes the image.

Segmentation algorithms for gray level images generally are based on one of two fundamental gray value properties. The first category looks for discontinuities in images based on large or abrupt changes in gray level values. Algorithms with this property look for isolated points, which differ from the surrounding, edges and lines in images. The second category of algorithms looks for similarities in the interesting objects and parts of images. Principal approaches are thresholding, region growing and region splitting and merging. Both segmentation categories will be discussed briefly below. For a more complete presentation, see Gonzalez [10].

4.3.1 Detection of Discontinuities

As mentioned above, the detection of discontinuities is based on abrupt changes in gray level values or isolated points. There exist three basic types of discontinuities, which are points, lines and edges. A very common and quite fast way of finding discontinuities is to run masks through an image.

Different masks are used to detect different discontinuities. For example, the general mask, shown in Figure 9, can be modified to detect points, edges and lines.

$$\begin{bmatrix} z_1 & z_2 & z_3 \\ z_4 & z_5 & z_6 \\ z_7 & z_8 & z_9 \end{bmatrix}$$

Figure 9: A general three by three mask.

In Figure 10, examples of masks used to detect points and lines are shown. These masks highlight the discontinuities while suppressing other parts of the image.

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}$$

Figure 10: Two three by three masks used for detecting isolated points (left) and horizontal lines (right).

4.3.2 Gradient and Laplacian of Images

When looking for discontinuities in images the gradient and the Laplacian of images become important tools. These properties are also used in many other situations.

The gradient of an image $f(x,y)$ at location (x,y) is the vector

$$\nabla \mathbf{f} = \begin{pmatrix} G_x \\ G_y \end{pmatrix} = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \\ \frac{\partial f}{\partial y} \end{pmatrix} \qquad (4)$$

An important quantity is the magnitude of the gradient. It is often referred to simply as the gradient and is given by

$$\nabla f = mag(\nabla \mathbf{f}) = (G_x^2 + G_y^2)^{1/2} \qquad (5)$$

In words, this quantity equals the maximum rate of increase of $f(x,y)$ per unit distance in the direction of $\nabla \mathbf{f}$. The gradient is often approximated with the computational faster absolute values, according to

$$\nabla f \approx |G_x| + |G_y| \qquad (6)$$

The partial derivatives $\partial f/\partial x$ and $\partial f/\partial y$ can be derived in many different ways. Here again, masks can be used to find these quantities. One of the common masks used is the Sobel operator. In Figure 11 the Sobel mask is given for the x and y direction, respectively.

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \begin{bmatrix} -1 & 0 & -1 \\ -2 & 0 & -2 \\ -1 & 0 & -1 \end{bmatrix}$$

Figure 11: The Soble masks used to compute the partial derivatives.

The Laplacian of a two-dimensional function $f(x,y)$ is a second-order derivative defined as

$$\Delta f = -\left(\frac{\partial^2 f}{\partial^2 x} + \frac{\partial^2 f}{\partial^2 y}\right) \qquad (7)$$

This quantity may also be implemented in numerous ways, as for the gradient above. A commonly used spatial mask is given in Figure 12. The Laplacian is used in many ways to detect edges and also to investigate image curvature.

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Figure 12: Most frequently used mask to compute the Laplacian.

4.3.3 Detection of Similarities

This approach and field of image segmentation algorithms includes thresholding, region growing and region splitting and merging. In this master's thesis only thresholding will be of interest and again the interested reader is referred to Gonzalez [10] for descriptions of the other techniques.

Thresholding is a very common approach to image segmentation. It is straight forward and easy to use in the standard case. But, in more difficult situations the thresholding technique is tricky and limiting.

Suppose an image has a histogram as the one given in Figure 13. A histogram is the frequency function $h(i)$ of an image $f(x,y)$, where i is the different gray levels representing the image. The histogram is of an image containing an object and a background. The object and the background consist of pixels partitioned into two different gray level groups. Each gray level group is normal-distributed and has a mean value quite different form the other.

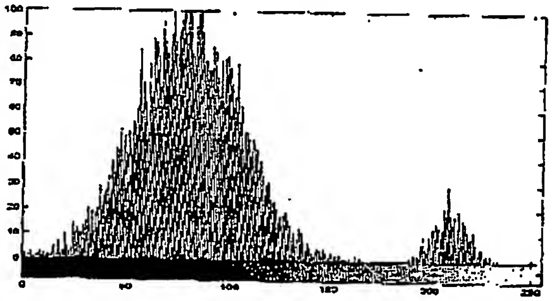


Figure 13: Histogram of an image with an object and a background.

By thresholding this image, the background and object are classified according to their intensities. This gives

$$g(x,y) = \begin{cases} 1 & if\, f(x,y) > T \\ 0 & if\, f(x,y) \leq T \end{cases} \qquad (8).$$

where pixels labelled 1 correspond to object and pixels labelled 0 correspond to background. In this case, with a histogram as in Figure 13, the threshold T should be set to 175. Doing so would classify the object and the background at a hundred percent accuracy.

4.3.4 Evolving Interfaces in Image Segmentation

In recent years a new approach to image segmentation has been developed. It is based on evolving interfaces, such as Snakes, see Kass [11], and other active contouring. The idea is to let a front grow from a starting seed, placed inside or outside the object to be segmented. By doing so, false noise boundaries due to artifacts can be avoided. In Figure 14 below, an example is given. In the image a white object pierced with small holes against a black background is shown. The desired segmentation result should be the larger outer boundary.

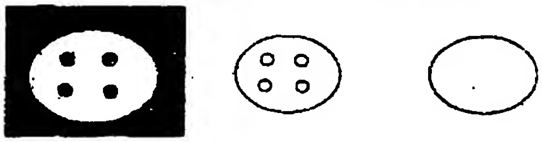


Figure 14: Three images showing the difference between thresholding and evolving interface segmentation. Left: Original image. Center: Segmentation with thresholding. Right: Evolving interface segmentation.

The two different approaches to image segmentation, shown in the center and the right image in Figure 14, illustrate the advantage with evolving interfaces versus thresholding. The power of the evolving interface approach, is that it has the ability to naturally execute topological changes and has good stopping criteria built on the gradient. With the help of the Fast Marching Method, introduced in Section 5, the evolving interface approach to image segmentation is a fast and very flexible segmentation method.

Recall, this master's thesis is based on segmentation with the help of evolving interfaces.

4.3.5 Image Segmentation Criteria

It is very important that the base for image analysis is founded properly. There exist several desirable properties which the segmentation has to satisfy to be a solid base to stand on. A well performing segmentation should be

1. *Fast*: Often the segmentation is one of many steps in a long chain during an image analyze. Therefore, it should be as fast as possible to reduce computational time.
2. *Repeatable*: It should be possible to repeat the same segmentation over and over again with the same result.
3. *Robust*: To get a correct and reliable result the segmentation need to be robust. The segmentation system also has to function properly even in unusual situation.
4. *Accurate*: It is necessary to have an accurate segmentation so that the segmented objects are as close to reality as possible.

The above stated criteria are sometimes difficult to satisfy all at a same time. For example, the segmentation is faster if it is less accurate. On the other hand it becomes less reliable. In common problems the most important criterion has to be satisfied at the cost of others. But, in some applications all criteria have to be satisfied to have a fully operational segmentation. These are very difficult segmentation problems and require highly specialized segmentation algorithms.

4.4 Image Modelling

Sometimes it is interesting to obtain gray level values in images between known intensity pixels. In the "real world" the objects represented by gray level values are continuous. In digital images they have been discretized. Thus, to get a value between a neighborhood of discrete pixels the value has to be interpolated from the neighbors. This could in some sense be called *image modelling.*

There exist several different interpolation algorithms with different properties, but here only the bilinear interpolation will be discussed. For other interpolation methods, please refer to Gonzalez [10].

To calculate the intensity in a point between discrete pixels, the bilinear interpolation uses gray level values of the four nearest neighbors to that point. The intensity in a point, p, is given by

$$I(p) = I_1cd + I_2ac + I_3bd + I_4ab \qquad (9)$$

where the intensities of the four known corners, placed in the centers of the surrounding pixels, are denoted $I_1 \cdots I_4$, and a, b, c and d are defined as in Figure 15.

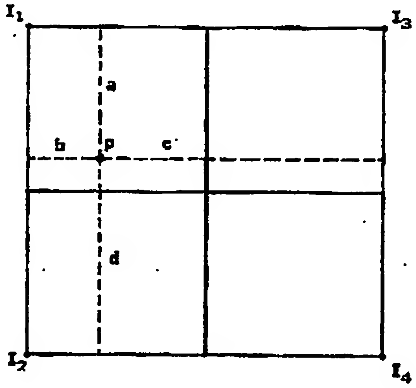


Figure 15: Bilinear interpolation.

5 Theory of Evolving Interfaces

5.1 Introduction

Evolving interfaces exist in varying settings in our surrounding. These are, among many, ocean waves, material boundaries and burning flames. Even though it seems strange, hand written characters, shapes against backgrounds and iso-intensity contours in images can also be described with moving interfaces, see Sethian [12].

A boundary is an example of an evolving interface. It can be described as a curve in two dimensions or a surface in three-dimensions. From here on, only the two-dimensional case is discussed, if nothing else is mentioned. Imagine that the curve separates the inside from the outside and that it moves outwards under the known speed function F. The curve expands in its normal direction. Below, in Figure 16, an example of a propagating boundary is shown.

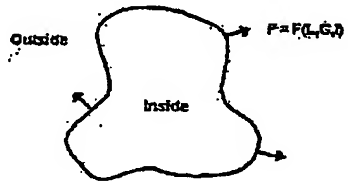


Figure 16: Propagating curve in normal direction with speed F.

The aim with evolving interface problems is to track the above curve as it evolves. First thing to do, in an attempt to describe the motion of the interface, is to find the speed function F. In a typical boundary formulation, the speed function might depend on the following factors

- *Local Properties (L).* These properties are controlled by the local information. They are, for example, normal vector and curvature.
- *Global Properties (G).* Global properties depend on the shape and position of the front. For example, the speed also depends on associated differential equations.
- *Independent Properties (I).* These are boundary shape independent and are decided by the underlying structure, in which the boundary is propagating. One example is a constant gradient that effects the movement of the interface.

Thus, the speed function F with the above stated properties can be written as

$$F = F(L, G, I) \tag{10}$$

The speed function determines the whole interface evolution. Therefore, it is very important that an adequate speed function is chosen to describe the boundary formulated problem.

To be able to track interfaces successfully, an Eulerian framework, see Sethian [12], must be set up. An Eulerian formulation is suitable because the underlying coordinate system remains fixed and does not vary in time.

5.2 Formulating the Differential Equation

One way of describing evolving interfaces is with the help of differential equations. There exist two main options to formulate the interface propagation as a differential equation. The first option is to let the speed function F always have values greater than zero. Then, the positions of the propagating front can be monitored by calculating its arrival time $T(x, y)$ at the position (x, y). The equation for the time function is easily found, because *distance = rate * time*, which gives the Eikonal differential equation

$$|\nabla T|F = 1, \qquad T = 0 \quad on \quad \Gamma \tag{11}$$

where Γ is the initial location of the interface.

Figure 17 below, shows the one-dimensional setup of the derived differential equation.

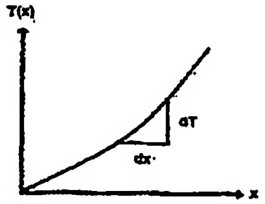


Figure 17: The one-dimensional setup of the derived differential equation (11).

Now the front motion can be characterized as the solution to a boundary value problem. If the speed function F depends on the position of the front, the differential equation will become a non-linear Eikonal equation.

The second option to formulate the differential equation does not demand that the speed function F is strictly positive. It is a more general formulation which allows the front to move past the same position (x, y) several times. When F is negative, the front can move backwards and revisit an already passed position. Hence, the crossing time function $T(x, y)$ is a multi-valued function. To describe the motion of the front, a higher-dimensional function ϕ with the fronts initial starting point as the zero-level set, must be introduced. This leads to the initial value formulation

$$\phi + F|\nabla\phi| = 0, \qquad given \quad \phi(x, t = 0) \tag{12}$$

In this master's thesis only the boundary value formulation will be of interest. The initial value formulation solved with numerical approximation schemes is called The Level Set Methods. Interested readers are referred to Sethian [12].

Returning to the boundary value formulation again, recall that the interface evolution will be known when the Eikonal equation (11) has been found. One problem in solving these equations is that the solution does not have to be differentiable, even with smooth initial boundaries. How can the solution to these differential equations solutions be found? One way is with the approximating numerical Fast Marching Method.

5.3 The Fast Marching Method

The Fast Marching Method is a computational technique, which approximates the solutions to the non-linear Eikonal equation of the form in Eqn. (11). The method also deals with non-differential solutions in a natural way, which otherwise could cause problems.

5.3.1 Non-Differentiability

To further investigate the solutions that might be non-differentiable, consider an example of a non-convex initial curve and its differential equation given by:

$$|\nabla T| = 1 \tag{13}$$

The speed function F is in this case constant and equal to one ($F = 1$). Suppose it is interesting to know where the interface is positioned after one unit in time. In Figure 18, the initial curve and its propagation in question are shown.

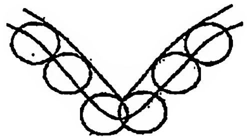


Figure 18: Solution to differential equation given in Eqn. (13).

The Above result shows that it is possible to end up with a non-differentiable solution even with a smooth initial boundary. This imply that the approximation schemes used for the Eikonal equation must be able to deal with non-differentiable solutions.

5.3.2 Approximation Scheme

In the Fast Marching Method, an approximation of the gradient using upwind differences is used. The upwind scheme is based on Euler's method, see Harris [13], for approximating the derivatives. By solving the ordinary differential equations outwards along the positive and negative x-axis, in the one-dimensional case, the approximation is given by

$$\frac{(T_{i+1} - T_i)}{\Delta x} = F_i \quad i > 0 \tag{14}$$

$$\frac{(T_{i+1} - T_i)}{\Delta x} = -F_i \quad i \leq 0 \tag{15}$$

where $T_0 = 0$. Hence, each ordinary differential equation is solved away from the boundary condition.

It is possible to approximate the Eikonal equation with an equation of motion given by

$$\psi_t = S(1 + \psi_x^2)^{1/2} \qquad (16)$$

in the one-dimensional case, where S is the speed function, see Sethian [14]. The spatial derivative ψ_x^2 can be approximated with a finite difference approximation built on the upwind scheme, see Osher [16]. This approximation yields

$$\psi_x^2 \approx (\max(D_i^{+x}\psi, 0)^2 + \min(D_i^{-x}\psi, 0)^2) \qquad (17)$$

where the standard notation for the finite difference given by

$$D_i^{-x}\psi = \frac{\psi_i - \psi_{i-1}}{h} \qquad D_i^{+x}\psi = \frac{\psi_{i+1} - \psi_i}{h} \qquad (18)$$

has been used. Above, ψ_i is the value of ψ on a grid at the point ih with grid spacing h.

Now, the Eikonal equation (11) can be written, in the two-dimensional case, as

$$|\nabla T| \approx \left(\begin{array}{c} \max(D_{ij}^{-x}T, 0)^2 + \min(D_{ij}^{+x}T, 0)^2 + \\ \max(D_{ij}^{-y}T, 0)^2 + \min(D_{ij}^{+y}T, 0)^2 \end{array} \right)^{1/2} = \frac{1}{F_{ij}}$$

where the forward and backward operators D^{-y} and D^{+y} are similar to the ones defined for the x-direction in Eqn. (18). A slightly different and more convenient approximation, see Rouy [17], is given by

$$|\nabla T| \approx \left(\begin{array}{c} \max(D_{ij}^{-x}T, -D_{ij}^{+x}T, 0)^2 + \\ \max(D_{ij}^{-y}T, -D_{ij}^{+y}T, 0)^2 \end{array} \right)^{1/2} = \frac{1}{F_{ij}} \qquad (19)$$

This equation is a piecewise quadratic equation for T_{ij}, given that the neighboring grid values for T are known. With this approximation the non-differentiability in Eqn. (11) is dealt with in a natural way and it will not cause any trouble.

The above approximations lead to information propagating from smaller values of T to larger values. Hence, the Fast Marching Method rests on solving the above equation by building the solution outwards from the smallest value of T.

By defining the building zone to a narrow band around the front it is possible to make the Fast Marching Method very fast, because it does not have to keep track of every value in the solution at the same time. The key to the speed lies in how to update the narrow band and how to select which grid point in the narrow band to update. Selecting the grid point to update is straight forward. The grid point with the lowest calculated T is the one to update.

5.3.3 Algorithm

The Above conclusions finally lead to the complete Fast Marching Method, which is, in the two-dimensional case

1. *Initialize Given Boundary:* Tag all grid points in the initial condition as *Known*. Continue by marking as *Trial* all unknown four-connected neighbors to the *Known* points. Finally, tag all other grid points as *Far*.

2. *Get Next Candidate Point:* Let the next candidate grid point be the one in *Trial* with the smallest value of T.
3. *Update Narrow Band:* Remove the candidate point from the set *Trial* and tag it as *Known*. Tag also all unknown four-connected neighbors of the candidate point as *Trial*. If a neighbor is of type *Far*, add it to the narrow band by removing it from the *Far* list and adding it to the set *Trial*.
4. *Recompute T Values:* Recompute the T values for all four-connected values of the candidate point in stage 2 by solving Eqn. (19).
5. *Loop:* While stopping criteria are not reached, go to stage 2.

In Figure 19 an example of the Fast Marching Method is given. The example is a boundary value formulation with known boundary value at the center grid point. Black grid points represent *Known*, gray *Trial* and transparent *Far* values.

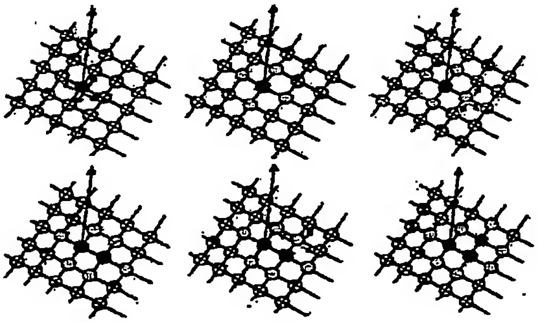


Figure 19: An example of a boundary value formulation solved with the Fast Marching Method. Each image represents a stage in the interface evolution. The center grid point is the initial boundary. Black grid = *Known*. Gray grid = *Trial*. Transparent grid = *Far*.

5.3.4 Updating Procedure on Orthogonal Mesh

A very important and central role in the Fast Marching Method is the updating procedure for T values. This is the procedure by which new trial values are created for T in the narrow band. By solving Eqn. (19) a new value of T is obtained. But how does this work in practice?

Solving Eqn. (19), means solving a piecewise quadratic equation for T_{ij}, assuming that the neighboring grid values for T are given. In the special case with an orthogonal mesh, an algebraic solution to Eqn. (19) can be found. In the following example this will be shown.

Imagine that an orthogonal mesh exists, like the one in Figure 20, below. Suppose that the goal with the updating procedure is to calculate a new value for T at the center grid point.

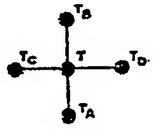


Figure 20: Example of an orthogonal mesh with five grid points.

The surrounding values of T are labelled according to $T_A = T_{x-1,y}$, $T_B = T_{x+1,y}$, $T_C = T_{x,y-1}$ and $T_D = T_{x,y+1}$. Standing at the center point, the Fast Marching Method attempts to solve the quadratic equation given by each quadrant. In this example only T_A, T_B and T_c act as contributors. The other grid point T_D is regarded as a *Far* value and does not have an influence on the solution. Thus, there are three cases to look at when solving the equation and they are

$\mathbf{T_A}$ is *Known*: The real solution with the smallest T so that $T > T_A$ is given by either

$$\begin{aligned}(T-T_A)^2 &= \frac{1}{F_{xy}^2}\\ &\text{or}\\ (T-T_A)^2+(T-T_B)^2 &= \frac{1}{F_{xy}^2}\\ &\text{or}\\ (T-T_A)^2+(T-T_C)^2 &= \frac{1}{F_{xy}^2}\\ &\text{or}\\ (T-T_A)^2+(T-T_B)^2+(T-T_C)^2 &= \frac{1}{F_{xy}^2}.\end{aligned}$$

$\mathbf{T_A}$ and $\mathbf{T_B}$ are *Known*: The real solution with the smallest T so that $T > T_A$ and $T > T_B$ is given by either

$$\begin{aligned}(T-T_A)^2+(T-T_B)^2 &= \frac{1}{F_{xy}^2}\\ &\text{or}\\ (T-T_A)^2+(T-T_B)^2+(T-T_C)^2 &= \frac{1}{F_{xy}^2}.\end{aligned}$$

$\mathbf{T_A}$, $\mathbf{T_B}$ and $\mathbf{T_C}$ are *Known*: The real solution with the smallest T so that $T > T_A$, $T > T_B$ and $T > T_C$ is given by

$$(T-T_A)^2+(T-T_B)^2+(T-T_C)^2 = \frac{1}{F_{xy}^2}$$

It is easily seen that the number of equations to be solved (M), related to the number of neighboring grid points with *Trial* values for T (n) is

$$M = 2^n$$

5.4 Efficient Sorting for the Fast Marching Method

To keep the Fast Marching Method fast, a god sorting technique must be utilized. The reason for this is that after every narrow band update, all *Trial* values in the narrow band have to be resorted. This is guarantied that the smallest value of T will be the next candidate. A heap sorting method is a very good method choice to keep track of the *Trial* values.

5.4.1 Heap Sort

Heap sort is an optimal way of sorting information. It does not require any extra space and has a time complexity of $O(nlog(n))$, which is very fast. The sorting technique is based on heaps. This data structure is explained in the following.

A heap is a tree where every node has a key more extreme (greater or less) than the key of its parent. The tree must be a complete tree, which means that

1. it is empty or
2. its left subtree is complete of height $h-1$ and its right subtree is completely full of height $h-2$ or
3. its left subtree is completely full of height $h-1$ and its right subtree is complete of height $h-1$.

Figure 21 below, shows an example of a max heap.

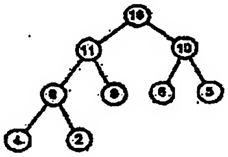


Figure 21: Example of a max heap with ten nodes.

In practice, a binary tree can be represented by a vector. This is a very efficient way of implementing a heap. The vector representation does not require any extra space if the tree is complete. By numbering the nodes from top to bottom and left to right, the children of node i are at $2i$ and at $2i+1$ if they exist, and the parent of node i is at $\lfloor i/2 \rfloor$ (integer part of the division) if it exists. Figure 22 shows a min heap and the equivalent heap vector.

To use the heap sorting algorithm in the Fast Marching Method certain operations on the heap must be supported. These are

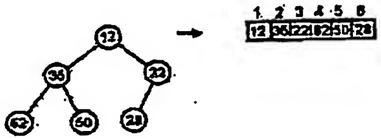


Figure 22: A min heap with six elements and its equivalent array.

1. Remove smallest value of T in the heap.
2. Add an element to the heap.
3. Update a key value at any given position in the heap.

These operations should be possible to complete while guaranteing that the heap remains proper. For further information please refer to Jeffrey [15]. It is also possible to extend the heap property to handle equal key values. The order of the equal key values is not guaranteed, but in this application it is irrelevant.

6 A Segmentation System

This section describes the implemented segmentation system applied on 2-DE images. Recall, by segmenting 2-DE images, means extracting as many protein spots as possible from the protein pattern in the 2-DE image. Below, in Figure 23 an overview of the system is given.

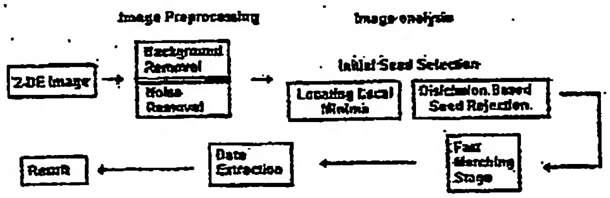


Figure 23: Overview of Segmentation System.

The system uses image preprocessing in an image enhancement stage to remove background variations and noise. Thereafter, an image analysis stage is performed. This is where the images are segmented and information about each segmented protein is stored in a structured way. In each of these stages many different tricks are used to find the optimal segmentation. The above introduced stages and substages are described in this section.

6.1 Background Variation Removal

Images received by the segmentation system almost always have varying backgrounds. Therefore, it is difficult to see differences between background and protein spots when segmenting several parts of the images at the same time. In one part the background might be very bright (high gray level values) and in such a part the proteins are also very bright. In other parts it is the reverse, with dark background and dark proteins. Figure 24 shows an example of the varying background effect.

This property makes it difficult to find similarities between proteins in the whole image, without including background pixels. By removing the background, a more uniform image is obtained, which makes it easier to separate background from proteins.

When removing the background the optimal result would be

1. Non-existing background or
2. Homogenous background with significantly different properties than the interesting proteins.
3. Controlled or no changes to the protein pattern visible in the images.

In this case only a coarse background removal was needed to improve the image segmentation result.

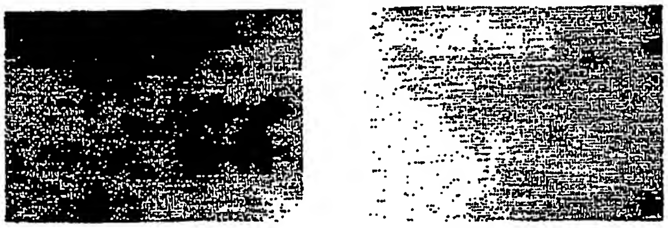


Figure 24: An example of the effect of varying background. Two different parts of the same image.

A simple and surprisingly well functioning background removal, based on bilinear interpolation, was created and used. The idea is based on the assumption that the background varies with low frequencies and does not have large discontinuities. This makes it possible to build a module of the background using bilinear surface blocks. To build the blocks, a bilinear interpolation method, according to Section 4.4, was used.

The following algorithm to modulate and subtract the background was used

1. *Decide Block Size:* The size of building blocks was decided based on the size of the image. A large image gives large block size.
2. *Divide the Image in Parts With the Same Size as the Blocks:* At the fringes the size of the image parts was chosen so that they, together with the other parts of the image, filled the whole image.
3. *Calculate Mean Intensity in Each Part of the Image:* In each part the mean intensity was calculated and stored in a vector.
4. *Take Mean Values to be the Corners of the Modulated Blocks*
5. *Create Modulated Blocks With Bilinear Interpolation Between the Corners for Each Block*
6. *Put the Modulated Blocks in a Background Image*
7. *Subtract Original Image With Background Image*

6.2 Noise Removal

Similar to background variations in the images, noise in the images is always present in varying amount. The reason to remove noise in the images is to make the initial seed selection, discussed in Section 6.3, easier.

Noise was removed in the images with a combination of a mean (ones filter) and a Gaussian filter, introduced in Section 4.2. The two filters were first convoluted with each other. Then the resulting filter was applied on the image. In this way only one convolution of the filter with the image was needed. Again, the filter sizes were determined by the size of the image.

6.3 Initial Seed Selection

One of the main issues in this segmentation system was to find good initial seeds. The initial seeds form a base for the evolving interface stage, discussed in Section 6.4. Each protein spot in the image should be marked with an initial seed. If a protein spot is not marked with a seed it will not be regarded as a protein. Instead it will be part of the background. In Figure 25, some proteins with corresponding initial seeds are shown.

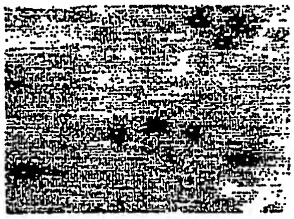


Figure 25: 2-DE image with initial seeds marking protein spots.

In 2-DE images the protein spots are represented by varying shapes and sizes of gray level valleys, see Figure 26. This protein spot property is used to find initial seeds that will mark each protein. Locating all local minima in the image and marking them as initial seeds is a good starting guess. A local minima identification on its own is not a sufficient criterion for protein spots. The use of some more protein spot specific characteristics have to be used to improve the seed guess. The Following three properties will be used to identify initial seeds

1. *Local Minimum*
2. *Curvature*
3. *Intensity*

They are all discussed, below, in the following sections.

6.3.1 Locating Local Minima

One way to locate local minima in images is to look at the gradient and the Laplacian. A local minimum is defined so that the gradient has a zero crossing and the Laplacian is negative. This approach works well in theory, but not in practice. In the discrete images which contain protein spot valleys, the gradient might never become zero. Also, the Laplacian might not be negative at the lowest value in the valley. Another approach is needed to find the local minima in the images.

Instead, the used method to identify local minima is based on pixel intensities. A local minimum is defined as a pixel with the intensity value, I, whose eight-connected pixels all have a value greater than or equal to I. In rare cases a local minimum region might get two or more markers. This happens when

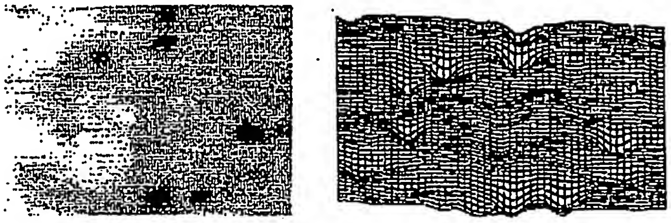


Figure 26: A 2-DE image with protein spots and its three-dimensional representation.

a ridge with higher values pushes up in the middle of the region. This is a desirable feature which increases the chances of protein detection. The following algorithm describes the method. If a whole region is a local minimum, the region only contains one marker.

1. *Get Center Pixel and its Eight-Connected Neighbors:* The fringe of the image is neglected.
2. *For Each Neighbor Look at Its Value:* The lowest neighbor value is stored in a variable.
3. *Neighbor has higher or Equal Value:* If neighbor is marked as local minimum unmark it.
4. *All Neighbors Have Higher or Equal Values:* Mark center pixel as local minimum.
5. *Loop:* While not end of image, go to Stage 1. (Walk through the image from right to left and top to bottom.)

6.3.2 Decision Based Seed Rejection

When locating local minima it often occurs that the local minima are situated in a background deviation, such as noise or artifacts in the image. To avoid treating these local minima as parts of a protein and using them as initial seeds, some decision based rejection method has to be used. A good way to find out if a local minimum is situated in a protein spot, is to look at the curvature in the local minimum surrounding.

The theory of finding a value of the curvature in an image was derived in Section 4.3.1, above. The curvature was calculated by finding the Laplacian of the image $f(x, y)$. A convex curvature is represented with values above zero and vice versa for the concave curvature.

Thus, a protein spot which has concave shape, has negative values in its local minimum surrounding. By defining the size of the surrounding and the amount of concaveness the protein spot has to show, a more robust initial seed identification can be done.

In some protein spots, it can occur that the spot is saturated, which yields that the bottom of the spot is cut off. By introducing a third criterion for the seed identifications, these kind of protein spots can also be identified, despite their non-concaveness. Figure 27 below shows the difference between an inverted saturated and non-saturated spot.

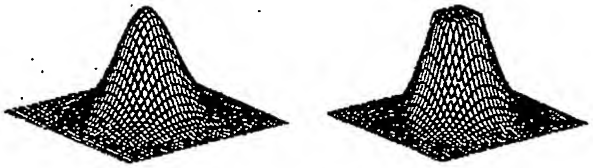


Figure 27: Inverted non-saturated (Left) and saturated (Right) spot.

6.4 Evolving Interface Stage

When the initial seeds that represent each protein spot have been located, the evolving interface stage is started. This stage is used to grow a region around each initial seed. After the region growing process is done, each seed has its own region associated with it and, thus, a part of the 2-DE image containing that specific protein type. This is how the segmentation is done.

The theory of evolving interfaces was introduced in Section 5. Evolution of the interfaces is based on the speed function delivered to the Fast Marching Method. The initial seeds have great impact on the result, but they have already been identified and are assumed correct, at this stage. Remaining, is the definition of a solid speed function and the stopping criteria that will halt the segmentation.

6.4.1 Defining the Speed Function

The definition of the speed function used by the Fast Marching Method is of great importance. Therefore, much effort has been used to create and identify the best speed function. A good speed function should have the following properties

1. High values inside protein spots.

2. Low or zero values elsewhere.

The reason to create a speed function with these properties, is to be able to get good stopping criteria for the interface propagation.

To define a good speed function, the nature of the protein spots have to be well know. Below, in Figure 28, a protein spot is shown in both one and two dimensions. As already known, the protein spot is represented by an intensity valley. Looking at the gradient might be a start. In Figure 29 the gradient of the protein spot, in Figure 28, is shown.

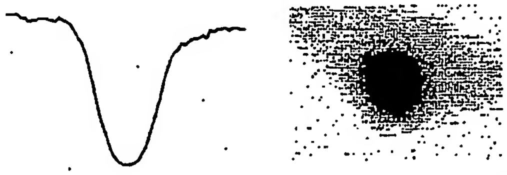


Figure 28: Protein spot in one and two dimensions.

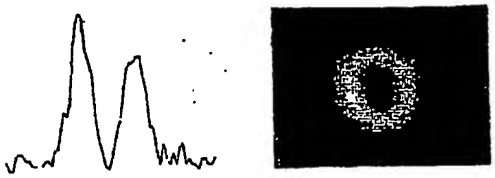


Figure 29: Gradient of protein spot in one and two dimensions.

By combining both the gradient and the intensity values from the image, a good speed function is obtained. The speed function is defined as

$$F(x,y) = \frac{1}{\alpha * e^{I_S(x,y)} * e^{G_S(x,y)}} \tag{20}$$

where α is a constant, $I_S(x,y)$ an intensity $(I(x,y))$ based function and $G_S(x,y)$ a gradient $(G(x,y))$ based function. It should be mentioned that this speed function is only of the type $F(I)$, where I is *independent properties*, explained in Section 5.1. This is the most simple type of speed functions and it does not take the *local* and *global* properties into account.

To be able to achieve the above properties for the speed function, $I_S(x,y)$ and $G_S(x,y)$ are created from each initial seed and summed outwards to the actual position (x,y). Below, in Figure 30, a schematic view of the creation of $I_S(x,y)$ and $G_S(x,y)$ is given. In the figure, K denotes *Known* values and T denotes *Trial* values. Non-marked positions are *Far* values. The speed in T_U is given by

$$F(T_U) = \frac{1}{\alpha * e^{I_S(T_U)} * e^{G_S(T_U)}}$$

where

$$\begin{aligned} I_S(T_U) &= I_S(K) + \frac{I_S(K_1) + I_S(K_2)}{2} + I(T_U) \\ G_S(T_U) &= G_S(K) + \frac{G_S(K_1) + G_S(K_2)}{2} + G(T_U) \end{aligned}$$

The idea is that for each step outward from the initial seed position, the value given is a summation of the mean values from the accepted neighbors to that point. The following algorithm shows the calculation of $I_S(x,y)$.

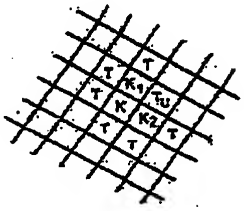


Figure 30: Schematic view over $I_s(x,y)$ and $G_S(x,y)$ creation. K denotes *Known* values. T denotes *Trial* values. Non-marked denotes *Far* values. Updating speed value in T_U.

1. *Get Point (x,y) for Which to Calculate $I_S(x,y)$:* This is simply the position in which the speed is to be calculated.
2. *Find Previous Values:* Identify all neighbors to the point (x,y) that are *Known*, se Section 5.3.3 for explanation, and take the mean of them.
3. *Add Previous Value With Current:* Take the mean value from Stage 2 and add it with the value in (x,y).
4. *Store new Value:* Store and return the value to the surrounding.

To improve the properties of the speed function even more, the gradient image G, on which $G_s(x,y)$ is based, has been manipulated. Imagine a very small protein spot. The gradient of the spot is not very high, even at the borders. Around the spot there is only a homogenous background with hardly any gradient at all. To acquire a correct speed function in such situations, the gradient image is multiplied with a factor to raise the gradient in the background regions.

Figure 31 shows the speed function for the above protein spot, shown in Figure 28. As can be seen, the speed function has full filled the properties that were desired.

Figure 31: Derived speed function for a protein spot in one and two dimensions $(\alpha = 1)$.

6.4.2 Stopping Criteria

The segmentation of the proteins has to stop when all of the proteins have been isolated. It is important to have good stopping criteria so that a robust and accurate segmentation is achieved.

Two well functioning stopping criteria in this segmentation system are based on the time evolution, $T(x, y)$ for the Fast Marching Method. When the proteins have been isolated, the speed of the evolving interface will decrease sharply and the time between each evolving step will increase. Thus, the time gradient will grow enormously an suites perfect as a stopping criterion. The total consumed time will also increase rapidly and works as a second stopping criterion.

A combination of these two criteria gives well defined halting condition for the segmentation.

6.5 Data Extraction

The final stage in this segmentation system is the data extraction stage. Here, the total intensity, position and curvature of all protein spots are stored in a text file.

7 Results

The evaluation of the segmentation system is done in three major parts. In the first part, test images are created and used to identify weaknesses in the system. Continuing, an evaluation is done with the use of real 2-DE data images. Finally, this segmentation system is compared with a software on the market.

The aim with this testing is to see if the system is able to handle all difficulties with 2-DE image segmentation, discussed in Section 2. The result presented here has not been generated with only one setting of the different changeable parameters in the segmentation system. The following parameters are possible to change

Gauss Filter Size The size of the Gauss filter used to remove noise in the images.

Mean Filter Size The size of the mean filter used to remove noise in the images.

Spot Curvature Threshold The curvature each local minimum surrounding has to have, to be regarded as an initial seed for a spot.

Spot Search Space The size of the surrounding to a local minimum, for which the curvature is calculated.

Background Filter Size The image is divided into blocks of this size. Each block generates a mean value to be used in the background modulating function. The larger filter size, the more coarse background module.

There exists an automatical parameter setting function in the system, but it was turned off during these tests. Instead, the parameters were manually optimized. With automatic values, the result would look very much the same.

Some figures in this section contain three-dimensional plots to illustrate results and intensity images. These plots have been inverted, so that a protein valley is instead represented by a mountain.

7.1 Testing Environment

Tests were performed on a PC with a PII 330 MHz processor and 384 MB internal memory. The environment used for tests and the implemented segmentation system were developed in Matlab 6.1. Several of the main features in the system were implemented with the Matlab External Interface and more specific, the C Mex-files.

7.2 Test Images

In this part several images were created to test the systems ability to deal with different difficulties. All the spots presented in the test images have been created with Gaussian properties according to Eqn. (2), given in Section 3.3. Below, four different topics are introduced and discussed.

7.2.1 Overlapping Spots

One of the major difficulties in segmenting 2-DE images is the overlapping spot problem. In Figure 32, two different images are shown and their respective segmentation result.

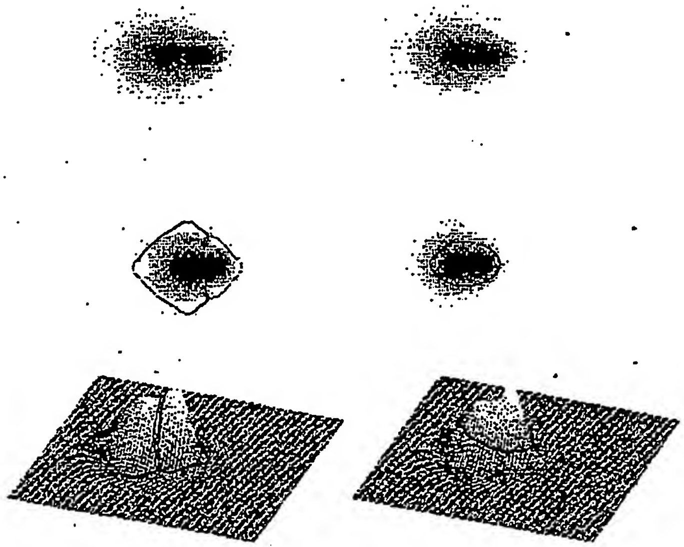


Figure 32: Overlapping spots. Left column: Two spots fairly separated. Right Column: Two spots too close.

The result above shows that two spots have to be enough separated, so that each spot has a local minimum associated with it. Otherwise, the segmentation will fail.

7.2.2 Noise in Images

It often occurs that the 2-DE images contain noise. Therefore, the segmentation system has to be non-affected by the noise. Below in Figure 33, two different spots are present in an image with applied noise. The noise is uniformly distributed on the interval (0.0,0.1). Observe that the image value varies between zero and one, which means that 10% noise was added.

The preprocessing noise removal takes care of the noise in a robust and accurate way. If the noise became too high typically 40% of the highest intensity,

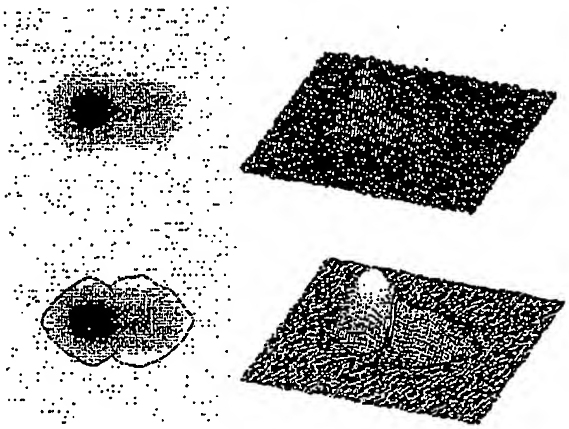


Figure 33: Two spots in an image with noise applied.

a correct segmentation was not possible. The reason was that under these circumstances faint spots drowned in the noise.

7.2.3 Images with Background

The background in the 2-DE images is sometimes very varying and the segmentation system has to perform well even in such situations. Figure 34, below, shows an image containing two spots with different size and a sinus-varying background given by

$$background(x) = A * sin(\varphi * x)$$

where $A = 0.3$, $\varphi = 0.1$ and $x = 1 \ldots 60$ was used.

By applying the background removal filter, described in Section 6.1, the effect of the background is minimized and the segmentation system has no problem with this kind of background. Again, with too much background variations, the segmentation failed, because faint spots diminished in the background. The amount of background variations needed for this situation to occur is rarely seen in real 2-DE images.

7.2.4 Different Sizes and Shapes of Spots

In 2-DE images the protein spots vary in size, shape and intensity. To see if the segmentation system can handle this diverse range of different spots, a test image with several spots was created. The result is presented below in Figure 35.

As can be seen by the result, the segmentation system is able to deal with different spot sizes, intensities, and shapes.

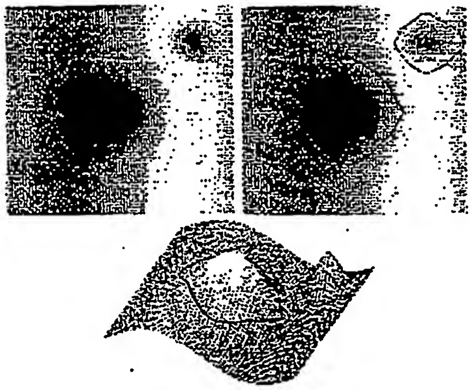


Figure 34: Two spots in an image with varying background applied.

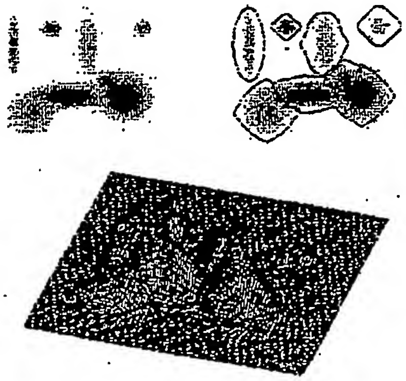


Figure 35: Test image with several different spots.

7.3 Real 2-DE Images

In this part, two real 2-DE images are segmented. The images and the result is shown in Figure 36.

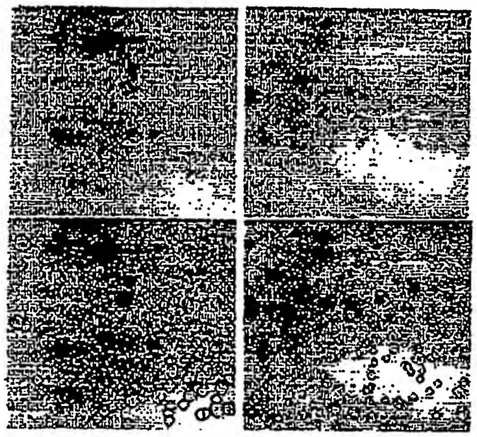


Figure 36: Two real 2-DE images that have been segmented.

About 320 protein spots were located in each image. All of them seemed to be correct identifications. In rare cases a protein spot was over segmented, due to characteristics in the 2-DE images. An example is the large protein spot in the left image located in the upper center.

It takes about 30 seconds to do a complete segmentation of an 2-DE image with a size of 1000 by 1200 pixels. Recall that much of the computations are done in Matlab, so if this system is implemented in a faster environment the time to perform a segmentation will be greatly reduced.

7.4 Comparison with Software on the Market

The segmentation system was compared with a software on the market. This software won a competition recently against other leading 2-DE analyzing softwares, which makes it a good competitor.

First, the result from the overlapping spots problem was compared, see Section 7.2.1. Secondly, the test image with noise and several different spots, in Section 7.2.4, was put to the test. Finally, the segmentation of two real 2-DE images were compared between the competitor and the implemented system in this thesis.

7.4.1 Overlapping Spots

The segmentation result is shown below in Figure 37. Both systems produces the same result.

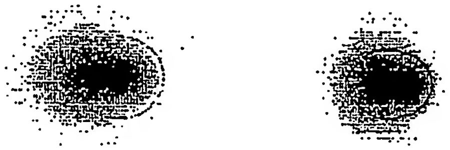


Figure 37: Two overlapping spots. Left: Competitor. Right: Implemented system.

7.4.2 Noisy Image with Different Spots

In Figure 38 below, the segmentation result is shown. The competitor is not able to find two of the spots, which the implemented segmentation system does.

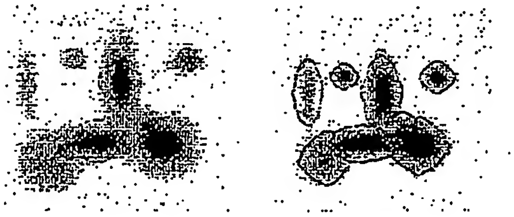


Figure 38: Noisy image containing several different spots. Left: Competitor. Right: Implemented system

7.4.3 Real 2-DE Images

In this comparison there is some difference between the results. In both cases the number of identified spots by the competitor were close to 260. This is a bit lower than the implemented system, which found around 320 spots. In Figure 39, the different results are shown.

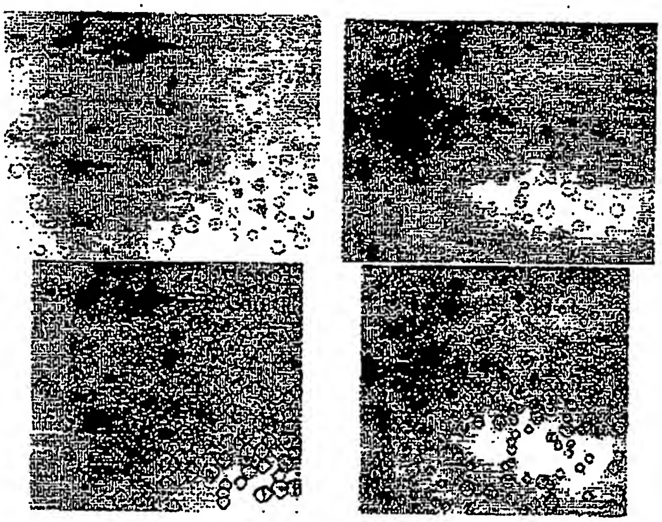


Figure 39: Segmentation result of two different 2-DE images. Upper: Competitor. Lower: Implemented system.

7.5 Formulating the methods used

- Method 1
A method for processing digital image data for two-dimensional gels by using Fast Marching Methods, comprising the steps of:
  - defining initial starting points for the methods;
  - generating a speed function for the methods;
  - generating interface propagations with said methods;
  - defining stopping criteria for the interface propagations for said methods;
  - generating image processing results based on the stopped said evolving interfaces.
- Method 2.
The method as recited in Method 1, wherein the starting points for said methods are generated from the said digital image.
- Method 3.
The method as recited in Method 1 or 2, wherein the speed function is dependent on said sample image intensities and functions thereof.
- Method 4.
The method as recited in Method 1-2 or 3, wherein the speed function is dependent on distances to said starting points and functions thereof.
- Method 5.
The method as recited in Method 1-3 or 4, wherein the speed function is dependent on said evolving interface curvatures and functions thereof.
- Method 6.
The method as recited in Method 1-4 or 5, wherein the speed function is dependent on said evolving interface normal directions and functions thereof.
- Method 7.
The method as recited in Method 1-5 or 6, wherein the speed function is dependent on said evolving interface positions and functions thereof.
- Method 8.
The method as recited in Method 1-8 or 9, wherein the speed function is dependent on said evolving interface shapes and functions thereof.
- Method 9.
The method as recited in Method 1, wherein the stopping criteria are dependent on said evolving interfaces time evolution and functions thereof.
- Method 10.
The method as recited in Method 1, wherein the stopping criteria are dependent on said evolving interfaces speed evolution and functions thereof.

- Method 11.
  A method for processing digital image data from two-dimensional electrophoresis gels by using Level Set Methods, comprising the steps of:
  - defining initial starting values for the methods;
  - generating a speed function for the methods;
  - generating interface propagations with said methods;
  - defining stopping criteria for the interface propagations for said methods;
  - generating image processing results based on the stopped said evolving interfaces.
- Method 12.
  The method as recited in Method 11, wherein the starting points for said methods are generated from the said digital image.
- Method 13.
  The method as recited in Method 11 or 12, wherein the speed function is dependent on said sample image intensities and functions thereof.
- Method 14.
  The method as recited in Method 11-12 or 13, wherein the speed function is dependent on distances to said starting points and functions thereof.
- Method 15.
  The method as recited in Method 11-13 or 14, wherein the speed function is dependent on said evolving interface curvatures and functions thereof.
- Method 16.
  The method as recited in Method 11-14 or 15, wherein the speed function is dependent on said evolving interface normal directions and functions thereof.
- Method 17.
  The method as recited in Method 11-15 or 16, wherein the speed function is dependent on said evolving interface positions and functions thereof.
- Method 18.
  The method as recited in Method 11-18 or 19, wherein the speed function is dependent on said evolving interface shapes and functions thereof.
- Method 19.
  The method as recited in Method 11, wherein the stopping criteria are dependent on said evolving interfaces time evolution and functions thereof.
- Method 20.
  The method as recited in Method 11, wherein the stopping criteria are dependent on said evolving interfaces speed evolution and functions thereof.

7.6 Benefits of Proposed Methods

The proposed methods are superior to other known methods for segmentation of 2-DE images. One of the most successful methods for segmentation of 2-DE images is described in Bettens [9]. The new methods described here, have a better time complexity and are more memory efficient. They are also easier expanded to include more complex dependencies by changing the property of the speed function $F(L, G, I)$. It is possible to let the evolving interfaces depend on the local curvature of the interface, the global shape of each segmented protein and other properties, for achieving good segmentation results, see Section 5.1. These method's generality leads to endless possibilities of improving the segmentation results.

8 Outlook and Further Improvements

The implementation of a 2-DE segmentation system seemed to be straight forward in the beginning of this master's thesis. But, as work advanced it showed that much efforts had to be done to achieve acceptable and reliable results.

Several important aspects of 2-DE image segmentation, such as varying background, noise removal and overlapping spots segmentation were just briefly touched and needed much more attention. Below, some further improvements are suggested.

Noise Removal With the help of more sophisticated filters and frequency analysis more of the noise effects could be removed. The Level Set Method, an evolving interface method, could be used to remove noise while keeping the interested contours intact. It has been used with great success in image preprocessing applications, see Sethian [12].

Varying Background Removal By investigating the histogram in different parts of the image a better value for the background modulation can be achieved. Also, the background modulation could utilize other more accurate interpolation methods, for example splines.

Marker Identifications The marker identification is the key to success in this segmentation system and more time should be spent on doing an even better identification of these. Again, different filters and perhaps spot models could be developed.

Create a More Complex Speed Function The speed function could be extended with curvature information and other properties so that the interface propagation could be curvature shape dependant.

Equilibrium Interface Propagation By allowing speed values to be both positive and negative an equilibrium segmentation could be possible. By doing so, the stopping criteria would be trivial.

Adjust Templates to Segmentation Result By defining certain templates that are suitable for protein spots, it might be possible to receive even better results.

Method for Digital Image Processing

CLAIMS

1. A method for processing digital image data for two-dimensionai gcls by using Fast Marching Methods, comprising the steps of:
   - defining initial starting points for the methods;
   - generating a speed function for the methods;
   - generating interface propagations with said methods;
   - defining stopping criteria for the interface propagations for said methods;
   - generating image processing results based on the stopped said evolving interfaces.

2. The method as recited in Claim 1, wherein the starting points for said methods are generated from the said digital image.

3. The method as recited in Claim 1 or 2, wherein the speed function is dependent on said sample image intensities and functions thereof.

4. The method as recited in Claim 1-2 or 3, wherein the speed function is dependent on distances to said starting points and functions thereof.

5. The method as recited in Claim 1-3 or 4, wherein the speed function is dependent on said evolving interface curvatures and functions thereof.

6. The method as recited in Claim 1-4 or 5, wherein the speed function is dependent on said evolving interface normal directions and functions thereof.

7. The method as recited in Claim 1-5 or 6, wherein the speed function is dependent on said evolving interface positions and functions thereof.

8. The method as recited in Claim 1-8 or 9, wherein the speed function is dependent on said evolving interface shapes and functions thereof.

9. The method as recited in Claim 1, wherein the stopping criteria are dependent on said evolving interfaces time evolution and functions thereof.

10. The method as recited in Claim 1, wherein the stopping criteria are dependent on said evolving interfaces speed evolution and functions thereof.

11. A method for processing digital image data from two-dimensional electrophoresis gels by using Level Set Methods, comprising the steps of:
    - defining initial starting values for the methods;
    - generating a speed function for the methods;
    - generating interface propagations with said methods;
    - defining stopping criteria for the interface propagations for said methods;
    - generating image processing results based on the stopped said evolving interface.

12. The method as recited in Claim 11, wherein the starting values for said methods are generated from the said digital image.

13. The method as recited in Claim 11 or 12, wherein the speed function is dependent on said sample image intensities and functions thereof.

14. The method as recited in Claim 11-12 or 13, wherein the speed function is dependent on distances to said starting points and functions thereof.

15. The method as recited in Claim 11-13 or 14, wherein the speed function is dependent on said evolving interface curvatures and functions thereof.

16. The method as recited in Claim 11-14 or 15, wherein the speed function is dependent on said evolving interface normal directions and functions thereof.

17. The method as recited in Claim 11-15 or 16, wherein the speed function is dependent on said evolving interface positions and functions thereof.

18. The method as recited in Claim 11-18 or 19, wherein the speed function is dependent on said evolving interface shapes and functions thereof.

19. The method as recited in Claim 11, wherein the stopping criteria are dependent on said evolving interfaces time evolution and functions thereof.

20. The method as recited in Claim 11 or 19, wherein the stopping criteria are dependent on said evolving interfaces speed evolution and functions thereof.

Method for Digital Image Processing

June 19, 2002

Abstract

Since the genome was sequenced, the importance of proteomics has increased enormously. To detect the different protein profiles contained in cells and other media, a two-dimensional gel electrophoresis method is used. It produces real and digital two-dimensional protein charts, which are analyzed. One of the first steps in the analysis is to detect the proteins in the digital chart. This is done with automatic computer-assisted segmentation.

In this master's thesis a segmentation system, based on evolving interfaces, is defined, implemented and evaluated for the segmentation of two-dimensional gel electrophoresis images. It is shown that with the use of Fast Marching Methods to approximate the evolving interfaces, the segmentation can be swiftly performed with high quality results. One weakness with this implementation is that it is a marker based segmentation. When two protein spots overlap and lay almost on top of each other, it is difficult to find correct markers. Fortunately, occurrences of this kind of extreme overlapping are rare.

Abstract

Since the genome was sequenced, the importance of proteomics has increased enormously. To detect the different protein profiles contained in cells and other media, a two-dimensional gel electrophoresis method is used. It produces real and digital two-dimensional protein charts, which are analyzed. One of the first steps in the analysis is to detect the proteins in the digital chart. This is done with automatic computer-assisted segmentation.

In this master's thesis a segmentation system, based on evolving interfaces, is defined, implemented and evaluated for the segmentation of two-dimensional gel electrophoresis images. It is shown that with the use of Fast Marching Methods to approximate the evolving interfaces, the segmentation can be swiftly performed with high quality results. One weakness with this implementation is that it is a marker based segmentation. When two protein spots overlap and lay almost on top of each other, it is difficult to find correct markers. Fortunately, occurrences of this kind of extreme overlapping are rare.